НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

____ _____ Теплоенергетичний факультет _____

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено

Завідувач кафедри

_____ О.В. Коваль
(підпис) (ініціали, прізвище)

“___”_____2019р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 “ Програмна інженерія “

на тему “Web-система для редагування та аналізу генетичних послідовностей“

Виконав (-ла): студент (-ка) ___4___курсу, групи_ТІ-51_

Жиров Максим Ігорович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник старший викладач Бандурка О. І. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

Напрям підготовки 6.050103 “ Програмна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019р.

ЗАВДАННЯ
на дипломну роботу студенту

_____
(прізвище, ім’я, по батькові)

1. Тема роботи _____“Web-система для редагування та аналізу генетичних послідовностей“_____
_____

керівник роботи_____Бандурка Олена Іванівна, старший викладач
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”____ 201__р. № __

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи _____Форма реалізації – веб-застосунок з інтерфейсом для користувача (HTML5, CSS, Angular) та серверна частина для нього (Java, Spring). Середовища розробки програмного продукту – Spring Tool Suite, IntelliJ IDEA, Visual Studio Code._____

4.Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)____Проаналізувати основні проблеми сучасної біоінформатики, дослідити існуючі програмні засоби для редагування та первинного аналізу генетичних послідовностей. Розробити веб-застосунок, що буде містити тимчасове сховище даних та зручний редактор для виконання базових операцій редагування та первинного аналізу генетичних послідовностей._____
_____
_____

_____

5. Перелік ілюстративного матеріалу

___«Мета розробки web-системи для редагування та аналізу генетичних послідовностей», «Функціональна модель системи», «Існуючі програмні рішення», «Паралельний алгоритм пошуку за зразком із фрагментацією та автоматичним сортуванням», «Програмні засоби реалізації», «Приклад структури серверної частини», «Обмеження на вхідні дані», «Приклади інтерфейсу користувача», «Висновки»_____

_____

_____

_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”_10_” ___жовтня_____2018р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	01.12.2018-31.03.2019	
2	Розробка архітектури та загальної структури системи	01-21.04.2019	
3.	Розробка структур окремих підсистем	22-28.04.2019	
4.	Програмна реалізація системи	29.04-13.05.2019	
5.	Оформлення пояснювальної записки	06.05-01.06.2019	
6.	Захист програмного продукту	25.05.2019	
7.	Передзахист	01.06.2019	
8.	Захист	17.06.2019	

Студент _____ Жиров М. І.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Бандурка О. І.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою дипломної роботи є створення ефективної web-реалізації засобів для редагування та первинного аналізу основних типів генетичних послідовностей.

Об’єктом дослідження є послідовності ДНК, РНК та протеїнів. Було виконано огляд існуючих програмних застосунків для редагування та аналізу генетичних послідовностей, виявлено їх переваги та недоліки. Створено web-систему, що вирішує деякі актуальні проблеми сучасної прикладної біоінформатики, зокрема пошуку за зразком.

Створена програмна система може бути використана у роботі науково-дослідницьких лабораторій у галузі генетики.

Загальний обсяг роботи: 51 сторінка, 20 ілюстрацій, 13 бібліографічних посилань та 3 додатки.

Ключові слова: генетичні послідовності, біоінформатика, паралельні алгоритми, пошук за зразком, web-система.

ABSTRACT

The goal of the work is creation of an effective web implementation for base genetic sequence types’ redacting and primary analysis tools.

DNA, RNA and protein sequences are the object of study. Review of existing program tools for genetic sequences’ redacting and primary analysis was conducted with their advantages and disadvantages summed up. Web system that solves a few of actual modern applied bioinformatics problems like pattern matching was created.

Developed software system can be used in a work of research laboratories in the field of genetics.

Total volume of the paper: 51 pages, 20 illustrations, 13 bibliography links and 3 appendixes.

Keywords: genetic sequences, bioinformatics, parallel algorithms, pattern matching, web system.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

1) ДНК — дезоксирибонуклеїнова кислота; речовина, що забезпечує збереження та передачу генетичної інформації.

2) РНК — рибонуклеїнова кислота; давніший у еволюційному сенсі аналог ДНК, виконує подібні функції.

3) Протеїн — органічна речовина, що відіграє важливі ролі у життєдіяльності живих організмів.

4) Web-система — розподілена система, що надає різним користувачам доступ до ресурсів мережі Інтернет.

ЗМІСТ

ВСТУП

Загроза енергетичної кризи ще у середині минулого століття спонукала вчених різних країн розпочати пошуки нових видів сировини, процес виділення енергії з яких би не викликав їх швидкого вичерпання та важких негативних екологічних наслідків для довкілля. Такі джерела енергії, в наш час широко відомі як відновлювальні, вже у близькому майбутньому зможуть витіснити традиційні. Але для цього все ще необхідно інвестувати у процес їх розробки суттєвий обсяг наукових та фінансових інвестицій.

Важливою тенденцією, характерною для напрямку розвитку сучасного наукового пошуку, є співпраця вчених різних дисциплін для досягнення потужних міжгалузевих рішень.

Найбільш потужні з точки зору наукових досліджень університети світу нерідко створюють міжнародні дослідницькі групи, до складу яких входять вчені різних спеціальностей та професій. Подібний підхід комплексної оцінки та вирішення проблем світової значущості без сумніву буде набувати все більшої популярності у майбутньому.

Пошуки ефективних відновлювальних джерел енергії все частіше звертають нашу увагу на біохімічний рівень організації живої матерії. Актуальною, як ніколи, постає проблема аналізу молекулярних процесів, що протікають у клітинах рослин, тварин, бактерій тощо.

Саме тому, біоенергетика як молодий та перспективний підхід до управління багатими енергетичними ресурсами не може існувати без наскрізної взаємодії з іншими науками про природу, що мають багатшу історію розвитку та великий накопичений досвід.

З іншого боку, виникає потреба обробки значних обсягів даних. Останні десятиліття можна охарактеризувати як тотальну інформатизацію не лише технічних, але і природничих наук. Одним з важливих наслідків цього складного

процесу є виникнення та стрімкий розвиток біоінформатики як нового методу реалізації дослідження живої матерії на молекулярному та клітинному рівні.

Таким чином, сучасні проблеми біоенергетики можуть бути ефективно вирішені лише із залученням новітніх біоінформатичних засобів обробки та аналізу інформації.

Оскільки біоінформатика є ще відносно новим науковим напрямком, існує велика потреба у розробці програмного забезпечення, яке б стимулювало його розвиток та полегшило пошук шляхів подолання розповсюджених проблем, які виникають у ході отримання, первинної обробки та подальшого всебічного аналізу найважливішої інформації про функціонування живого організму – його генетичного коду.

Генетичний код у вигляді послідовностей протеїнів та РНК і ДНК, які їх кодують, міститься у кожній клітині будь-якої живої істоти. Він регулює процеси утворення нових клітин та їх елементів, реакції виділення та розщеплення хімічних сполук.

Завдяки розумінню структури генетичного коду конкретного живого організму, можна отримати досить чітке сприйняття цілісної картини біохімічних реакції, які в ньому протікають, в тому числі тих, які є цікавими з точки зору біоенергетики.

Ефективне дослідження даного коду, втім, неможливе без використання новітніх програмних засобів, які автоматизують існуючі алгоритми його обробки та первинного аналізу, сприяють появі нових та дозволяють оцінити правильність результатів.

Таким чином, біоінформатика та біоенергетика є важливими складовими набору дисциплін, які сьогодні використовуються для вирішення надважливої проблеми збільшення обсягів та якості вироблення одного з видів відновлювальної енергії, що має на меті поліпшення екологічної ситуації в світі та подолання потенційної енергетичної кризи, яка може бути викликана вичерпанням традиційної енергетичної сировини.

Задача розробки сучасних апаратних та програмних засобів, що могли б спростити вирішення окремих біоінформатичних задач, цілком очікувано постає перед розробниками, які переслідують мету створення та впровадження забезпечення, що відіграватиме важливу роль у активному процесі дослідження актуальної науково-технічної проблематики.

Потреба у такому забезпеченні постійно постулюється учасниками міжнародних конференцій, які обговорюють сучасний стан розвитку галузі та можливі шляхи пришвидшення темпів розвитку різноманітних напрямків, що просувають цілу біологічну науку до майбутніх визначних звершень.

Потреба у попередній біологічній освіті часто відлякує сторонніх розробників для входження на профільний ринок. Однак, практика показує, що розробники широкого профілю, за умови консультацій із фахівцями з предметної області, можуть створювати достатньо якісний продукт, який потім може бути використаний безпосередньо даними фахівцями або науковими лабораторіями, інтереси яких вони представлятимуть.

Взаємодія між розробниками спеціалізованого програмного забезпечення та дослідниками галузі є плідною, коли обидві сторони володіють чітким уявленням про причини та цілі співпраці. Формування такого уявлення неможливе без проведення міжнародних зустрічей з метою створення та налагодження співробітництва як між науково-дослідницькими лабораторіями великих університетів, так і між компаніями, бізнес-інтереси котрих перетинаються з предметними областями біоінформатики та біоенергетики.

1. ЗАДАЧА СТВОРЕННЯ WEB-СИСТЕМИ РЕДАГУВАННЯ ТА АНАЛІЗУ ГЕНЕТИЧНИХ ПОСЛІДОВНОСТЕЙ

Була поставлена задача створення мережевого додатку, що повинен мати базовий функціонал у вигляді набору інструментів для первинного аналізу та обробки генетичних послідовностей трьох різних типів, а саме РНК, ДНК та протеїнів.

Система повинна містити у собі сховище даних для тимчасового збереження певного набору послідовностей. Тільки користувач з особливими правами доступу адміністратора отримає засоби для безпосередньої роботи з даним сховищем. Неавторизовані користувачі системи, втім, будуть мати можливість вільного використовування усіх інструментів додатку.

1.1. Задача розробки ефективного набору інструментів для редагування та первинного аналізу генетичних послідовностей

Інтенсивний розвиток біоінформатики протягом кількох десятиліть сприяв активному формуванню великої кількості проблем та шляхів їх вирішення. Система, яку потрібно розробити, в будь-якому випадку не зможе надати інструментарій для вирішення абсолютно всіх відомих задач даної галузі.

Відповідно було обрано ефективний підхід, який передбачає створення базового набору інструментів з можливістю легкого додавання нових засобів редагування та первинного аналізу послідовностей у майбутньому.

Це можна буде досягти за рахунок відділення клієнтської частини представлення результатів та отримання команд на виконання від безпосередньої реалізації вирішення задач редагування та первинного аналізу генетичних послідовностей.

Було вирішено створити самостійний сервіс для кожного типу послідовностей, оскільки певні з вище зазначених проблем мають зміст лише при роботі з конкретним типом. Однак більшість інструментів є доступною для кожного сервісу, що сприяє уніфікованому сприйняттю всієї системи при комплексному підході, який передбачає одночасну роботу з більш ніж одним типом послідовностей.

У якості проблем для базового набору інструментів були обрані наступні:

— видалення помилкових символів з послідовності;

— транскрипція ДНК;

— генерація випадкової послідовності для вказаної кількості символів;

— генерація комплементної (оберненої) ДНК;

— вимірювання відсоткового співвідношення пар нуклеотидів у ДНК;

— мутація послідовності;

— пошук подібних послідовностей у базі.

Найскладнішим завданням можна вважати останню з наведених проблем.

Необхідно було надати засоби, які б суттєво полегшили роботу дослідника із системою, а саме

— легке завантаження послідовностей до вікна редагування та бази даних;

— повернення до результатів досліджень, проведених раніше;

— зворотне повернення, якщо здійснюється перегляд архівних результатів роботи сервісів системи.

Очікується, що вхідні дані будуть заноситися до системи у форматі, який передбачає чергування рядку опису з рядком, що безпосередньо містить елементи генетичної послідовності як великі латинські літери.

Для вирішення даних задач було обрано комплексний підхід, який передбачає як використання існуючих алгоритмів, так і розробку власних там, де ефективні алгоритми все ще створені, або не є загально доступними.

Фактично вирішення кожної задачі полягає у роботі з текстовою інформацією, яка має неявну структуру, що дозволяє використовувати алгоритми загального призначення, в тому числі сортування, перевірки входження елементів у масив, генерації випадкових чисел з проміжку.

Після створення системи необхідно оцінити ефективність її роботи як в цілому, так і для вирішення конкретних проблем. Це можна виконати, наприклад, шляхом порівняння швидкодії створеної системи зі швидкодією аналогів, наявних у вільному доступі в мережі.

Очікується, що для методів вирішення різних задач буде досягнуто різний ступінь ефективності, оскільки їх розробка буде виконуватися незалежно, за невеликим винятком, коли один із методів буде потребувати попередню обробку цільової послідовності іншим (наприклад, видалення помилкових символів).

1.2. Задача створення багатокомпонентної web-системи

Оскільки певні частини системи, яку планується розробити, можуть бути у подальшому використані як компоненти для інших систем, необхідно у процесі розробки притримуватися принципів розподіленої мікросервісної архітектури, тобто окремі частини програмного застосунку повинні взаємодіяти між собою як незалежні програми, кожна з яких виконується на своєму сервері.

Пропонується створити 3 такі складові:

— сховище даних;

— реалізація інструментарію редагування та аналізу послідовностей;

— взаємодія з користувачем системи у середовищі web-браузера.

Система повинна бути зручною у використанні як для розробників, що будуть займатися її доповненням або інтеграцією до інших систем, так і для кінцевих користувачів достатньо широкого спектру.

Система повинна відрізнятися швидкодією, оскільки буде призначена для роботи з великим обсягом даних. Саме тому, взаємодія між різними компонентами системи повинна протікати без затримок, створюючи для кінцевого користувача своєрідну ілюзію монолітної системи.

Система повинна мати 2 типи користувачів: відвідувач та адміністратор. Адміністратор, на відміну від простого відвідувача, повинен мати доступ до функціоналу керування базою послідовностей системи, що передбачає:

— видалення послідовностей певного типу;

— додавання послідовностей певного типу з файлу, що має стандартний формат.

Розподіл компонентів сценарію використання системи за двома ролями вказано на діаграмі прецедентів (рисунок 1.1).

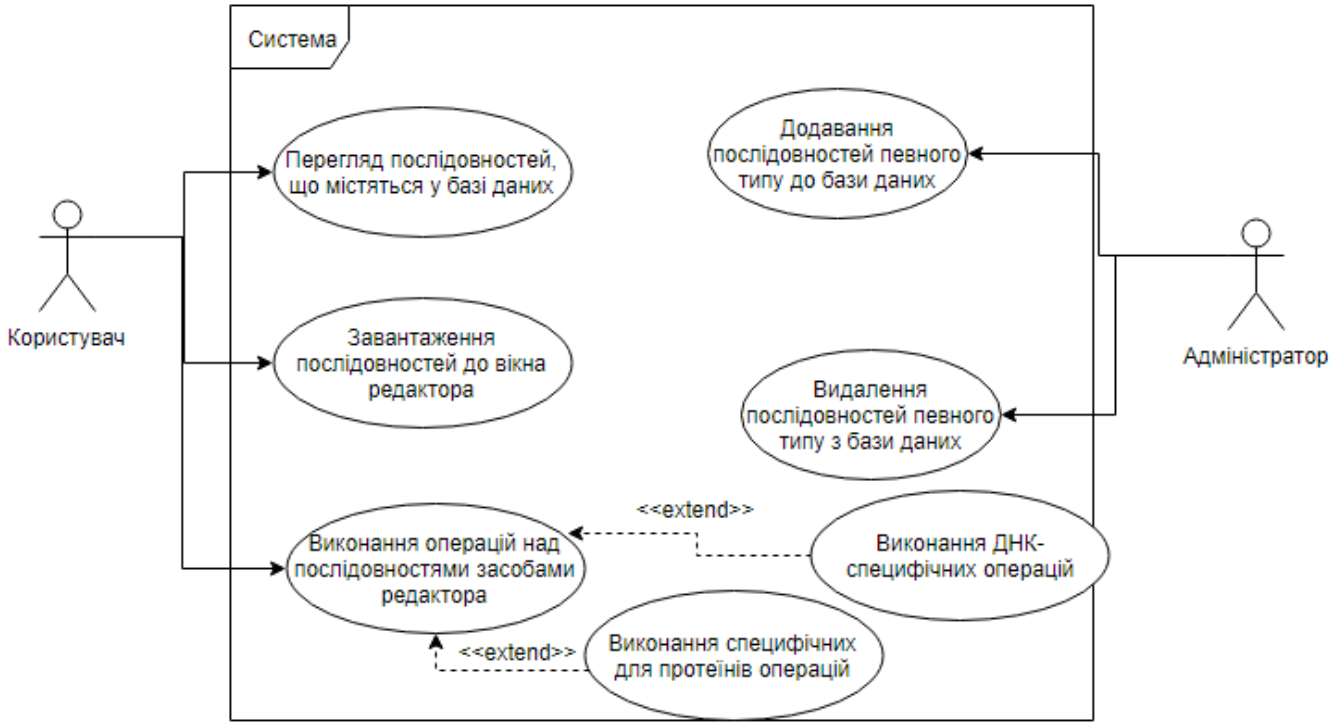

Рисунок 1.1 — Діаграма прецедентів системи

Відвідувач отримає можливість виконувати всі види операцій зі своїми послідовностями без можливості їх збереження у базі даних системи.

1.3. Висновки до розділу

Отже, було сформульовано задачу створення багатокомпонентного мережевого застосунку для виконання операцій редагування та первинного аналізу генетичних послідовностей заданого формату.

2. АНАЛІЗ ПРОБЛЕМИ РЕДАГУВАННЯ ТА АНАЛІЗУ ГЕНЕТИЧНИХ ПОСЛІДОВНОСТЕЙ

Біоінформатика як наукова галузь має потужну систему визначень, оволодіння якою є необхідне для чіткого розуміння предметної області.

Генетичні послідовності, в свою чергу, є одним з найважливіших предметів досліджень біоінформатичної науки, оскільки кодують інформацію, яка регулює центральні біохімічні процеси у клітинах та живих організмах в цілому.

2.1. Біоінформатика

Обчислювальна молекулярна біологія (біоінформатика) — це поєднання біології, а саме таких її розділів як генетична інженерія, молекулярна біологія та біохімія, з інформатикою.

Дана наука є досить молодою. Вважається, що вона виникла у кінці 1980-х років 20-го століття, коли людство домоглося суттєвого покращення існуючих чисельних методів обробки великих обсягів інформації різної структури. Біологічні дисципліни саме потребували використання таких методів для вирішення накопичених проблем.

Подібно до того як винайдення мікроскопічної техніки спонукало науково-технічний прогрес у сфері цитології (науки про структуру та функції живої клітини), побудова потужної обчислювальної техніки надала необхідні технічні засоби для розвитку генетики.

Біоінформатика займається проблемами збереження, перетворення та тлумачення різноманітних біоданих (даних, утворених у результаті всебічного вивчення живої матерії).

З іншого боку, біоінформатика вивчає молекулярні правила та системи, що керують, впливають на структуру, функції та еволюцію різноманітних форм життя [1].

Ця наукова галузь спирається на чисельні методи багатьох математичних та технічних дисциплін, в тому числі

— прикладної математики;

— статистики;

— теорії алгоритмів і структур даних.

2.2. Генетичні послідовності

Генетичні послідовності конкретного живого організму часто називають його генетичним кодом, що утворює певні паралелі між програмуванням та біологією та доводить важливість розуміння особливостей їх структури для оцінки та прогнозування процесів, що протікають у даному організмі.

Жива матерія може бути як клітинною, так і позаклітинною (віруси, віроїди, пріони, вірусоїди). Оскільки не всі вчені вважають позаклітинні форми життя насправді живими, є сенс розглядати клітину як найменшу повноцінну фізичну та функціональну одиницю організації живої істоти.

Клітини, в свою чергу, можуть містити ядро у явному (еукаріотичні) або у неявному (прокаріотичні) вигляді. Генетичний апарат усіх еукаріотів надійно захищений ядерною оболонкою. Взагалі, еукаріотичні клітини мають складнішу будову, ніж прокаріотичні (рисунок 2.1)[2].

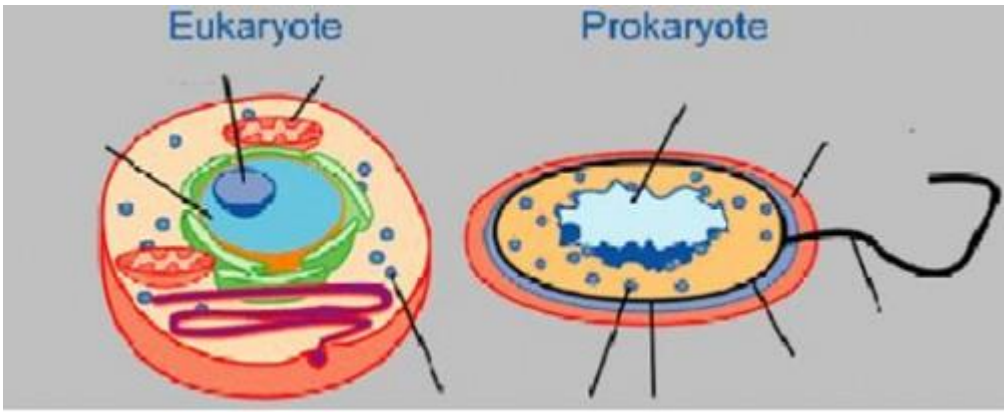

Рисунок 2.1 — Порівняння будови еукаріотичних (Eukaryote) та прокаріотичних (Prokaryote) клітин

У будь-якій клітині механізми поділу, росту, функціонування регулюються генетичною інформацією, яка представлена послідовністю нуклеїнових кислот (РНК для прокаріотичних та ДНК для еукаріотичних організмів).

Розглянемо сутність генетичної послідовності на прикладі ДНК еукаріотичної клітини.

Дезоксирибонуклеїнова кислота (ДНК) — це дуже довга полімерна молекула, що складається з простих елементів (нуклеотидів). Кожен нуклеотид, у свою чергу, складається з трьох складових:

— фосфатної групи (азотистої основи);

— пентозного (рибозного) цукру;

— основи: аденіну (A), гуаніну (G), цитозину(C) або тиміну(T).

У складі рибонуклеїнової кислоти (РНК) замість тиміну зберігається урацил (U).

Аденін та гуанін є похідними пурину. Цитозин та тимін — пірімидину.

Кожна молекула ДНК складається з двох пов’язаних спіралей. При цьому, якщо на конкретному місці в одній із спіралей знаходиться аденін, то на цьому ж місці в іншій спіралі повинен знаходитися тимін. Ідентичний зв’язок характерний для пари гуанін — цитозин.

Для зручності сприйняття послідовність ДНК записують як послідовність великих латинських літер, що позначають основи нуклеотидів, в напрямку від 5’ до 3’.

Згідно твердження, відомого як центральна догма, в еукаріотичній клітині відбуваються процеси транскрипції та трансляції.

Транскрипція — це процес перетворення ДНК у РНК.

Трансляція — це процес перетворення РНК у послідовність протеїнів.

Протеїн — це органічна сполука, що складається з амінокислот 20 різних типів. Проміжок РНК для трансляції починається зі стартового триплету (AUG) та закінчується стоп-триплетом (CAA, CAG або UGA). Всі інші триплети, що знаходяться між ними, кодують відповідні амінокислоти.

Таким чином, ДНК та РНК кодують відповідні протеїни.

Фенотип — це поєднання генетичної інформації організму та умов довкілля, в яких він існує.

У свою чергу, геномом живого організму називають всі ДНК послідовності, що містяться у клітинах даного організму.

Геном людини містить 24 хромосоми (22 аутосоми та 2 статеві). Третя частина геному, при цьому, кодує протеїни [1].

Проект визначення геному людини (The Human Genome Project) був активним з 1990 до 2003 року та обробка його результатів досі не досягла свого завершення. Однак необхідно відмітити великий прогрес, який було досягнуто у даній області за рахунок великої кількості різноманітних досліджень, що були виконані фахівцями різних галузей міжнародних університетів та наукових лабораторій (рисунок 2.2).

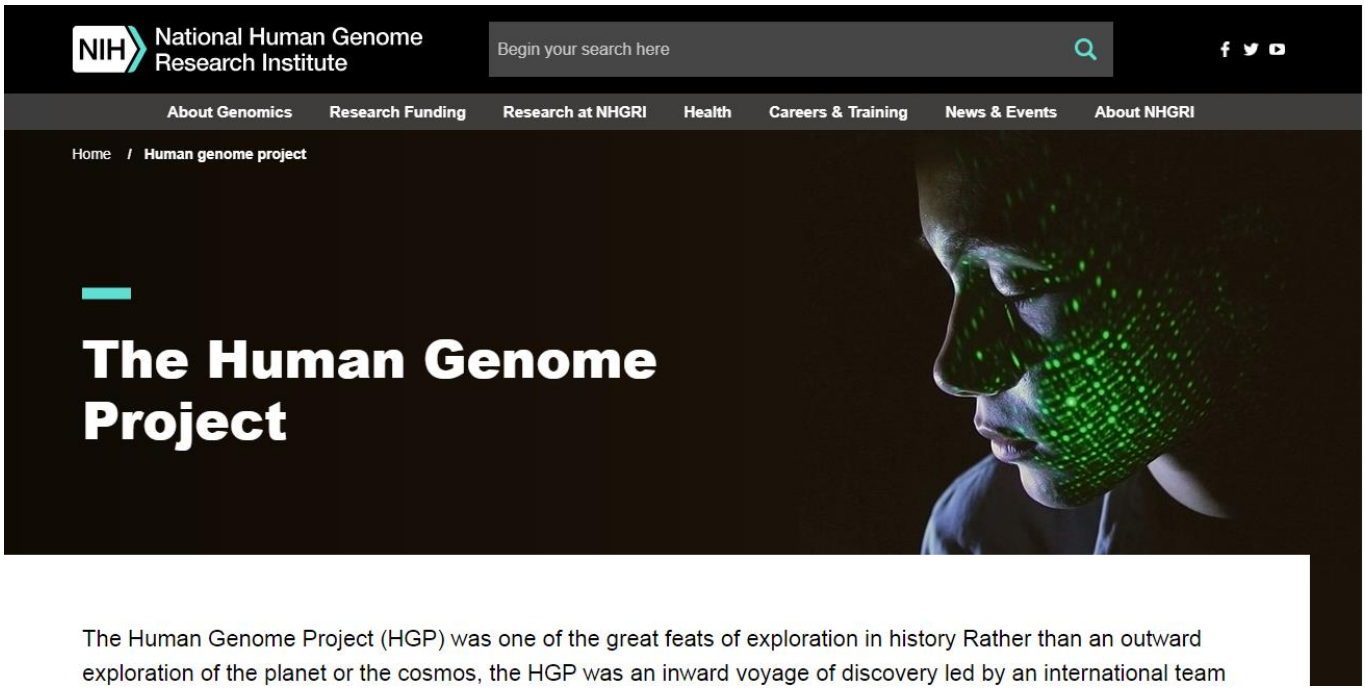

Рисунок 2.2 — Сторінка Національного дослідницького інституту геному людини (США), присвячена Проекту геному людини

Інтерпретація отриманих у ході виконання проекту результатів є однією з найбільш актуальних проблем сучасної інформатики, оскільки може сприяти суттєвому прогресу у розумінні ключових процесів діяльності організму людини. Однак існують й інші проблеми, поверхневий огляд яких виконано у наступному розділі.

2.3. Проблеми сучасної біоінформатики

Актуальність біоінформатики для сьогодення та її цілісна складність сприяє виділенню з неї окремих дисциплін. Наприклад, протеїнова інженерія займається ідентифікацією протеїнів. При цьому використовуються методу електричного заряду та молекулярних мас.

Для класичних розділів біоінформатики також є характерною достатня кількість проблем без загально-прийнятих рішень. Однією з таких проблем є пошук у базі послідовності, подібної до заданої. Проблема існує, бо постає питання як

розмістити одну послідовність по відношенню до іншої, щоб їх можна було порівняти.

Іншою проблемою є пошук мотиву (закономірності повторень) у послідовності.

Актуальною галуззю є компаративна генетична інженерія, що займається побудовою філогенетичних (еволюційних) дерев.

Для великих послідовностей все ще не існує остаточного вирішення проблем їх редагування та відтворення мутацій основ:

— заміни;

— вставки;

— вилучення.

Задача знаходження подібних до цільової послідовності зразків у великій базі даних завжди була надзвичайно актуальною для біоінформатичних програмних комплексів різної функціональної складності. Зазвичай, кожен розробник такого комплексу створює власну реалізацію вирішення цієї задачі, враховуючи особливості конкретної предметної області, завдання з якої він намагається вирішити.

Складність проблематики, що вивчається, можна показати на прикладі задач структурної біоінформатики. Дана галузь займається вивченням просторової структури протеїнів. Для цього використовується, наприклад, рентгеноструктурний аналіз.

Для розв’язання подібних проблем використовуються не тільки класичні числові методи, але також методи статистики та машинного навчання.

Універсальність подання первинної структури генетичного коду у вигляді текстової послідовності великих латинських літер дозволяє застосувати до неї практично всі відомі загальні алгоритми пошуку, сортування, ефективної модифікації тощо. Зокрема, для вирішення задачі швидкого сортування, враховуючи вище зазначену специфіку, може бути ефективно застосоване сортування підрахунком. Для послідовностей нуклеотидів, оскільки мова йде лише про чотири

можливих елементи (ключі), її використання дає ще кращі результати у порівнянні з протеїновими послідовностями амінокислот.

Існування настільки великої кількості програмних засобів у біоінформатиці може помилково змусити думати, що всі тривіальні задачі галузі вже давно вирішені, а для побудови програмних систем, що б мали змогу вирішувати найскладніші, потрібні міцні знання у галузі молекулярної біології та генетики, однак насправді існування великої кількості подібних програмних рішень спричинене іншими двома факторами:

— величезною популярністю та актуальністю досліджень у предметній області;

— відсутністю універсальних та остаточних рішень для цих, на перший погляд, простих задач.

Отже, стрімкий сучасний розвиток біоінформатики потребує від розробників різноманітних програмних систем максимальної співпраці з фахівцями предметної області, залучення вже існуючих теоретичних та практичних відкриттів у галузі та бажання інвестувати свій час у науково-технічний прогрес людства.

2.4. Існуючі програмні рішення

2.4.1. Sequence Manipulation Suite: Version 2 від Bioinformatics.org

Досить широко відомою системою, що містить функціонал для здійснення стандартних операцій редагування та аналізу генетичних послідовностей, є web-набір від Bioinformatics.org (рисунок 2.3) [3].

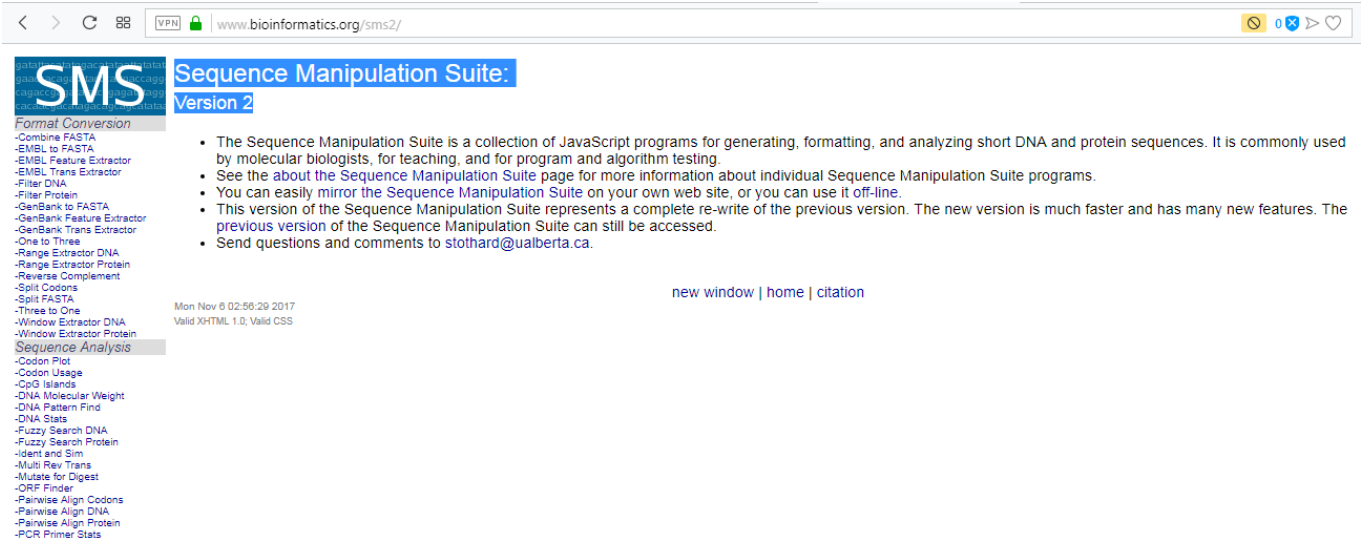

Рисунок 2.3 — Головна сторінка системи Sequence Manipulation Suite

Попри широкий набір інструментів, суттєвими недоліками даної системи є:

— використання застарілих технологій, що не дозволяє отримати максимально можливу швидкодію при обчисленнях;

— незручний інтерфейс користувача, що не дозволяє легко обрати потрібний інструмент і швидко ним оволодіти.

2.4.2. Cybertory від attotron.com

Засіб Cybertory (рисунок 2.4) [4] надає певний функціонал, який можна використовувати для елементарного аналізу генетичних послідовностей, супроводжує кожен інструмент детальною інструкцією, однак також є відчутно застарілим рішенням.

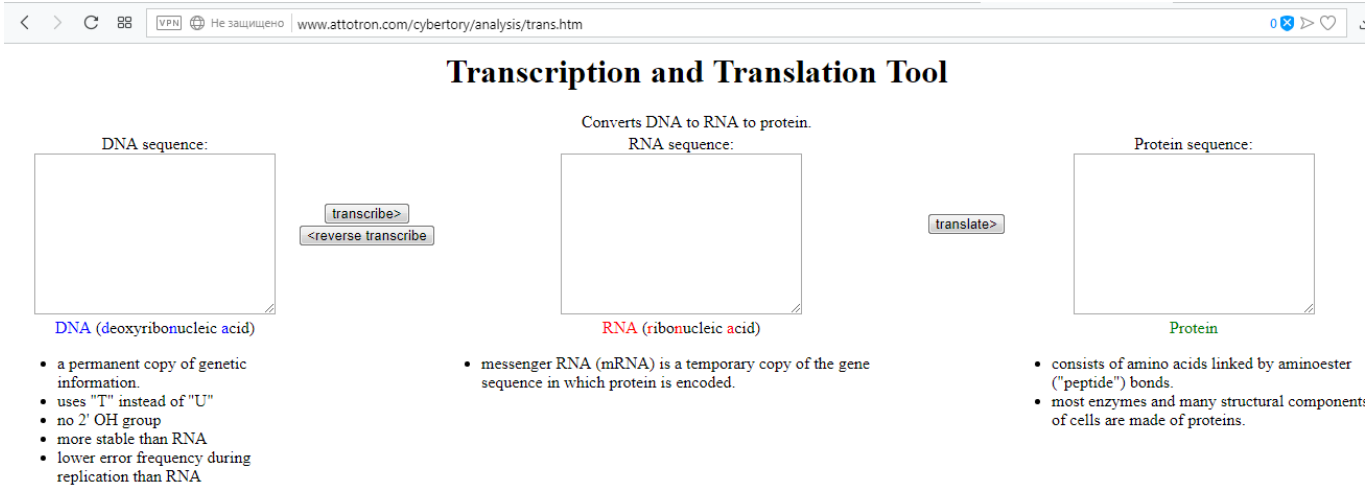

Рисунок 2.4 — Сервіс транскрипції та трансляції системи Cybertory

Іншим суттєвим недоліком даного рішення є обмежений набір інструментарію для аналізу генетичних послідовностей.

2.4.3. BLAST від National Center for Biotechnology Information

Засіб Basic Local Alignment Search Tool (рисунок 2.5) [5] є загально відомим та популярним рішенням проблеми пошуку подібних послідовностей до заданої.

Програмний продукт Національного центру біотехнологічної інформації (США) працює як з протеїнами, так і з послідовностями нуклеотидів, має низку програмних інтерфейсів для взаємодії з іншими засобами обробки біоданих.

Недоліком системи можна назвати відсутність можливості здійснення пошуку у базі даних, створеній користувачем.

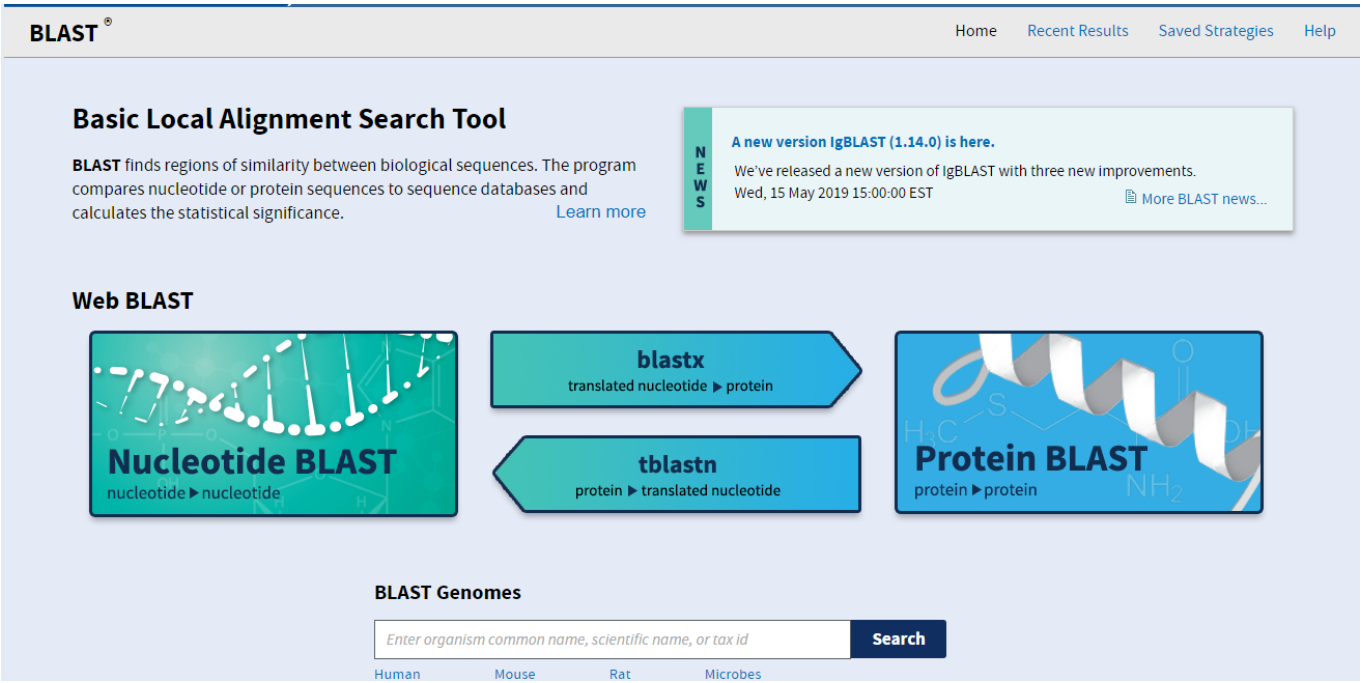

Рисунок 2.5 — Інтерфейс засобу BLAST

2.5. Висновки до розділу

Отже, було описано основні поняття предметної області та існуючі програмні рішення для виконання редагування та первинного аналізу інформації, що міститься у генетичних послідовностях.

3. ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

Для створення програмного продукту було використано широкий спектр засобів різного призначення.

3.1. Сховище даних

Для тимчасового збереження бази послідовностей у системі було обрано NoSQL базу даних MongoDB [6], що розподілена на кластері Amazon.

3.1.1. MongoDB

Під базою даних розуміють дані та правила їх збереження. Якщо потреби перших обчислювальних машин задовольняли прості текстові файли, то сучасні програмні засоби потребують більш серйозних рішень, які б дозволили вирішувати питання ефективного доступу до ресурсів, від яких критично залежить робота цілої програмної системи.

Стандартом для баз даних є реляційна модель, що групує дані у таблиці та використовує мову запитів SQL (Structured Query Language) для взаємодії з ними через інтерфейс системи управління базами даних.

Бази даних NoSQL (тобто ті, що не використовують стандартну реляційну модель таблиць і зв’язків між ними) відрізняються більшою швидкодією, однак не дозволяють проводити транзакції, що трохи зменшує їх показник надійності.

Оскільки база даних системи є лише тимчасовим сховищем, показник надійності не є критичним, на відміну від швидкодії. Саме тому було обрано NoSQL систему, де дані зберігаються у вигляді JSON-документів.

Стандарт JSON — це розповсюджений формат даних, призначений для їх передачі через мережу. Дані передаються у вигляді масиву мови JavaScript. Необхідно уникати використання службових слів мови у якості ключів, оскільки

деякі засоби для роботи з даними у форматі JSON можуть обробляти їх некоректним чином.

Оскільки єдиною істотною проблемою документно-орієнтованих баз даних є їх ненадійність, для виправлення цього недоліку було вирішено розподілити екземпляр MongoDB на кластері Amazon WebServices.

Даний хмарний сервіс надає надзвичайно надійні засоби для

— розгортки сховища даних на потужній обчислювальній техніці;

— аналізу статистики використання бази даних;

— моніторингу дискового простору та швидкості виконання операцій доступу до даних.

Проект MongoDB Atlas (рисунок 3.1) дозволяє налаштувати розподілений екземпляр бази даних MongoDB на кластері, постачальником ресурсів для якого може бути як Amazon WebServices, так і інша служба. За замовчуванням кожна база даних має список IP-адрес, які можуть до неї підключатися, однак даний функціонал можна відключити, що не рекомендується.

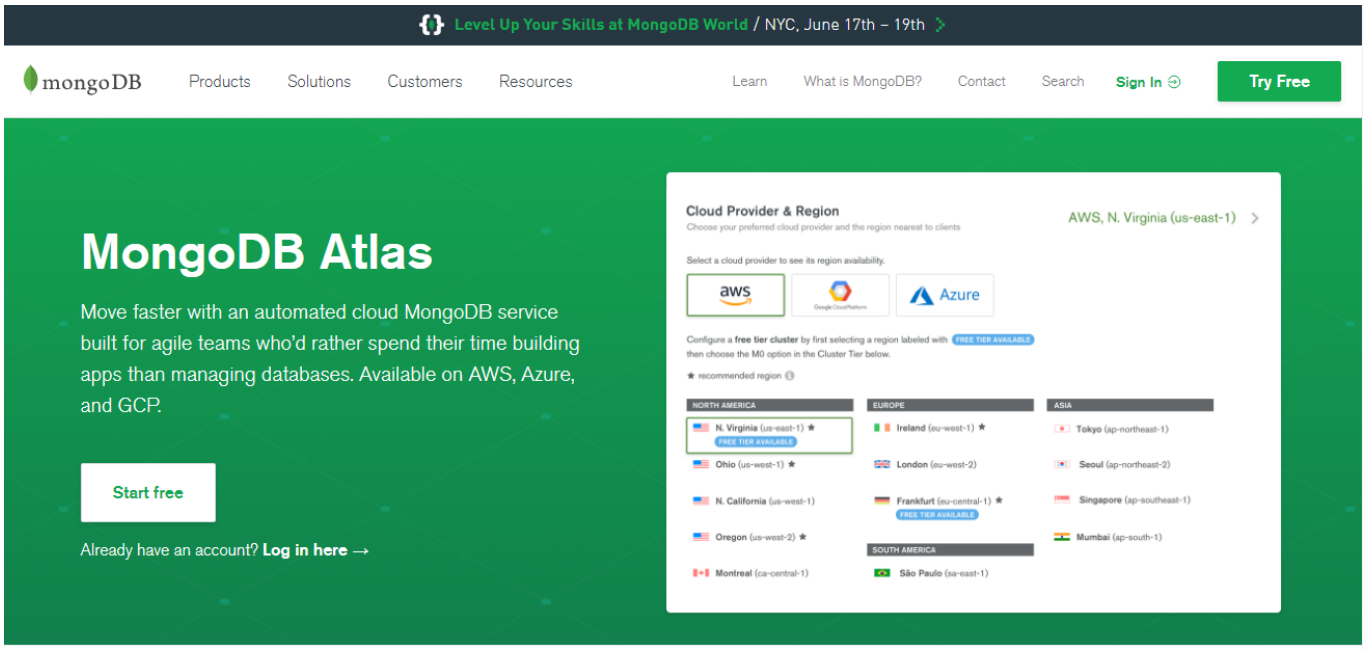

Рисунок 3.1 — Головна сторінка проекту MongoDB Atlas

Важливо зазначити механізм Replica set, що використовується для підвищення надійності системи сховища даних, оскільки усі дані одночасно існують у декількох копіях та не можуть бути легко втрачені, оскільки кожна така копія є фізично

незалежною. Повна втрата даних можлива лише за умови одночасного фізичного збою усіх потужних серверів. Тобто, головний недолік NoSQL баз даних, їх ненадійність як наслідок відсутності транзакцій, повністю виправляється механізмом розподілення даних у системі Mongo Atlas.

Для доступу до екземпляру бази даних формується рядок підключення, що містить дані про користувача та сховище.

3.2. Серверна частина

Серверна частина системи була реалізована на мові програмування Java з використанням наступних фреймворків:

— Maven — для керування налаштуваннями проекту;

— Spring Boot — для швидкого та спрощеного налаштування проекту;

— Spring MVC — для комунікації з клієнтською частиною з використанням стандартного шаблону “модель — представлення — контролер”;

— Spring Data — для автоматичної взаємодії зі сховищем даних, незалежно від особливостей реалізації останнього.

3.2.1. Maven

Рішення Apache Maven [7] — фреймворк для автоматизації управління Java-проектами.

Всі налаштування відбуваються у файлі формату XML (eXtensive Markup Language), який називається об’єктною моделлю проекту (рисунок 3.2).

Рисунок 3.2 — Вигляд XML-файлу налаштувань проекту

Використання цього засобу полегшує процес управління зовнішніми бібліотеками, версіями та безпосереднім запуском проекту.

Існують аналогічні рішення Apache Ant та Gradle.

Засіб Apache Ant є достатньо старою утилітою для виконання автоматичної збірки проектів на мові програмування Java. До його недоліків можна віднести імперативну модель налаштування, що означає потребу у написанні більшої кількості коду в порівнянні з декларативним стилем Apache Maven.

Засіб Gradle використовує скриптову мову Groovy замість XML-конфігурації. Це означає необхідність вивчення додаткової мови для виконання задачі правильного налаштування проекту. Навіть зважаючи на те, що синтаксис Groovy дуже подібний до синтаксису мови Java, формат конфігурації, що використовується у Apache Maven можна охарактеризувати як простіший та зручніший для вивчення і сприйняття.

3.2.2. Spring Boot

Рішення Spring Boot [8] — проект, що містить набір найбільш розповсюджених стартових конфігурацій для різноманітних Java-проектів.

В залежності від призначення проекту, можна підібрати відповідний пакет зовнішніх бібліотек та стандартних налаштувань (рисунок 3.3), що пришвидшує початковий етап розробки архітектури складної системи.

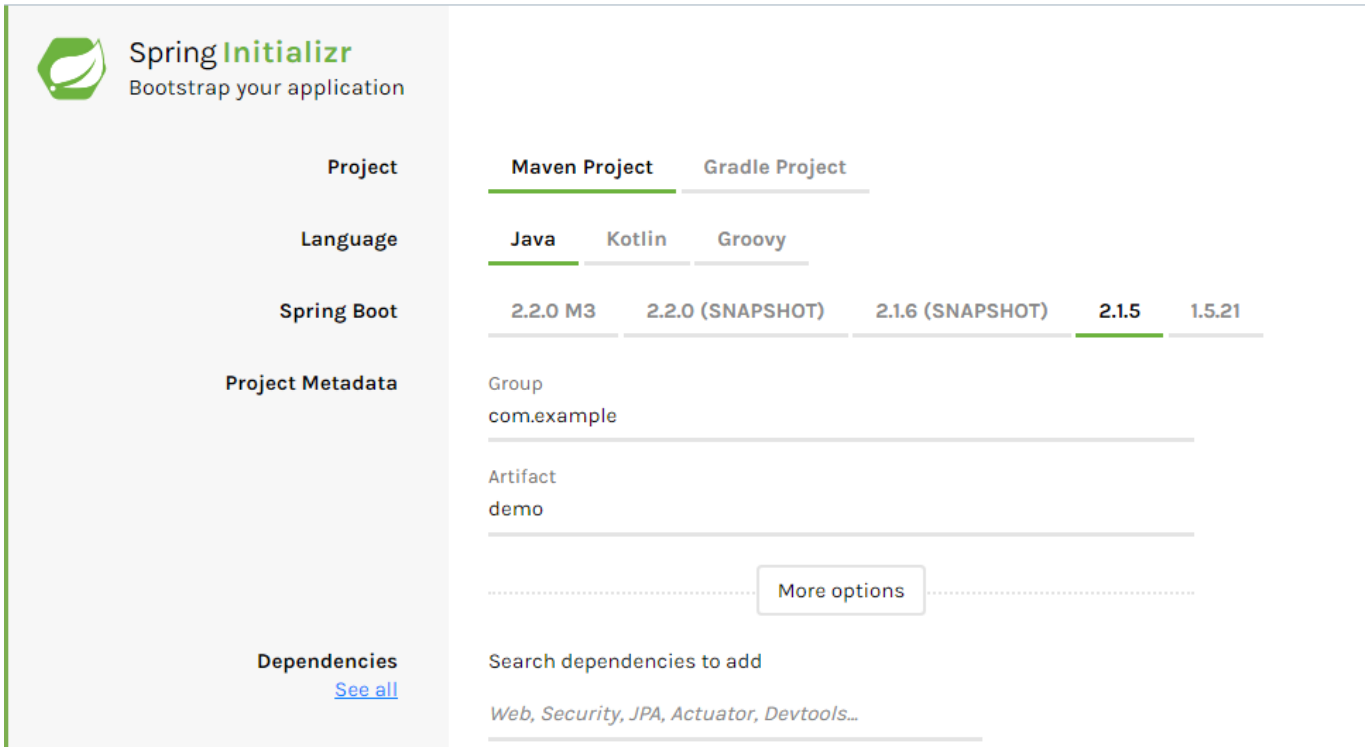

Рисунок 3.3 — Створення нового проекту за допомогою Spring Boot

На даний момент є можливість обрати пакет налаштувань у наступних категоріях:

— Core — основні бібліотеки;

— Web — бібліотеки для побудови web-систем;

— Template Engines — засоби побудови шаблонів web-сторінок;

— Security — засоби захисту систем від несанкціонованого доступу;

— SQL — бібліотеки для роботи з реляційними базами даних;

— NoSQL — бібліотеки для роботи з деякими іншими сховищами даних;

— Messaging — бібліотеки для реалізації механізму повідомлень у системі;

— Cloud — бібліотеки взаємодії з хмарними технологіями;

— I/O — засоби логування (виведення діагностичної інформації) подій;

— Ops — засоби діагностики функціонування системи на сервері.

Стартовою точкою проекту, створеного за допомогою ініціалізатору Spring Boot, є клас, що містить анотацію @SpringBootApplication. Одна анотація дозволяє замінити певний обсяг шаблонного коду, який є однаковим для майже усіх застосунків, оскільки визначає параметри їх запуску.

Стратегією проекту Spring Boot є створення пакету бібліотек та засобів, який вирішує близько 90% задач, що найчастіше зустрічаються у налаштуванні для кожного з основних типів проектів на мові програмування Java, надаючи розробнику можливість самостійного налаштування для найважливіших 10%, що безпосередньо залежать від цілей застосунку.

3.2.3. Spring MVC

Рішення Spring MVC — фреймворк для створення Java web-проектів з простою та універсальною структурою.

В основі фреймворку лежить ідея контролеру, який отримує запит від клієнтської частини, відповідно до якого виконуються необхідні дії у моделі системи, а результат повертається до клієнтською частини через представлення або у форматі JSON для REST API.

Методи контролеру обробляють HTTP-запити кінцевого клієнта. Протокол HTTP (HyperText Transfer Protocol) визначає комунікацію між програмами (зокрема, браузером клієнта та сервером) у мережі Інтернет. Найчастіше використовуються чотири метода HTTP:

— GET — отримання ресурсу із сервера;

— POST — створення нового ресурсу на сервері;

— PUT — оновлення існуючого ресурсу на сервері;

— DELETE — видалення ресурсу на сервері.

Стандарт HTTP є лише рекомендацією, однак його зазвичай притримуються розробники web-систем, оскільки уніфікований підхід сприяє побудові кращої та зручнішої архітектури для застосунків.

Метод та адреса у фреймворку Spring MVC передаються до методів контролера через анотацію @RequestMapping, а не задаються в окремому конфігураційному файлі роутингу, як в деяких інших подібних рішеннях, що спрощує процес написання програмного коду.

3.2.4. Spring Data

Проект Spring Data [9] — зручний фреймворк для взаємодії Java-проектів зі сховищами даних.

Фреймворк автоматично створює сервіс для певного класу моделі, який дозволяє виконувати з об’єктами цього класу базові операції бази даних. При цьому налаштування бази даних (в тому числі драйвера) задаються у налаштуваннях проекту.

Проект містить модуль для роботи з базою даних MongoDB, що містить

— можливості конфігурації для екземпляру сховища даних та драйвера;

— службовий клас MongoTemplate, що підвищує продуктивність найчастіше використовуваних операцій по відношенню до MongoDB;

— підтримку подій життєвого класу сутностей моделі;

— можливості низькорівневого відображення тощо.

3.3. Клієнтська частина

Клієнтська частина системи була реалізована на мові програмування TypeScript з використанням фреймворку Angular 6 та мов розміток HTML і CSS.

3.3.1. TypeScript

Мова TypeScript — це надмножина популярної скриптової мови програмування JavaScript, створена компанією Microsoft.

Особливістю мови є типізація, що попереджує появу великої кількості помилок, характерних для класичного JavaScript.

Програмний код на мові програмування JavaScript, в цілому, буде коректним для інтерпретатора мови TypeScript, однак деякі транслятори вимагають від розробника використання строгої типізації.

3.3.2. Angular 6

Рішення Angular 6 [10] — фреймворк для розробки клієнтської частини web-систем (рисунок 3.4).

Особливостями даного фреймворку в порівнянні до інших подібних є:

— орієнтація на компонентне програмування;

— використання TypeScript замість класичного JavaScript;

— велика база бібліотек доповнень, розроблених активними користувачами;

— зручна документація.

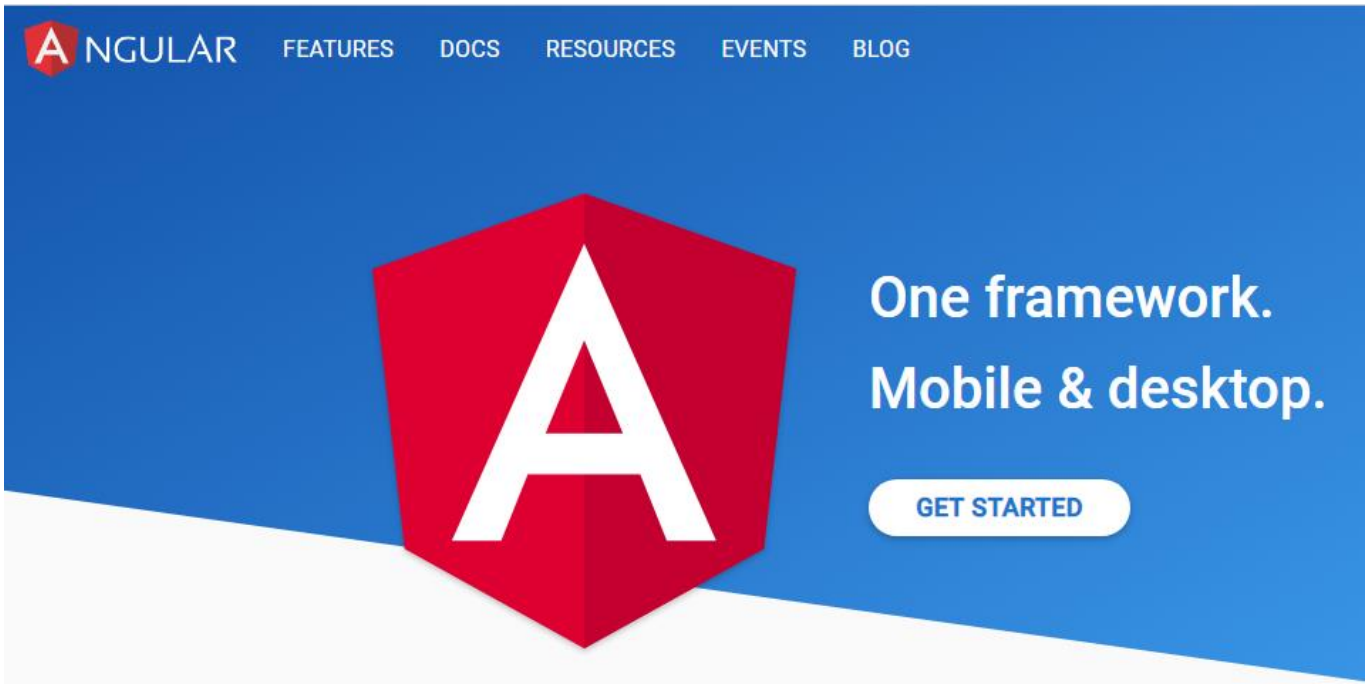

Рисунок 3.4 — Головна сторінка проекту Angular

Кожен проект, створений за допомогою цього фреймворку, має подібну структуру.

Компонентна парадигма програмування реалізована у вигляді компонентних пакетів, кожен з яких містить 4 файли:

— розмітка HTML;

— стилі CSS;

— програмний код на мові TypeScript;

— модульні тести для компоненту.

3.4. Середовища розробки програмного забезпечення

Система була створена у інтегрованих середовищах розробки Spring Tool Suite, IntelliJ IDEA та Visual Studio Code.

3.4.1. Spring Tool Suite

Середовище Spring Tool Suite [11] — створена на базі середовища Eclipse система для легкого налаштування та розробки Java-проектів з використанням Spring-фреймворків.

3.4.2. IntelliJ IDEA

Середовище IntelliJ IDEA [12] — одне з найпопулярніших інтегрованих середовищ розробки для мови програмування Java (рисунок 3.5).

Середовище надає можливість встановлення різноманітних плагінів (доповнень), які спрощують процес розробки.

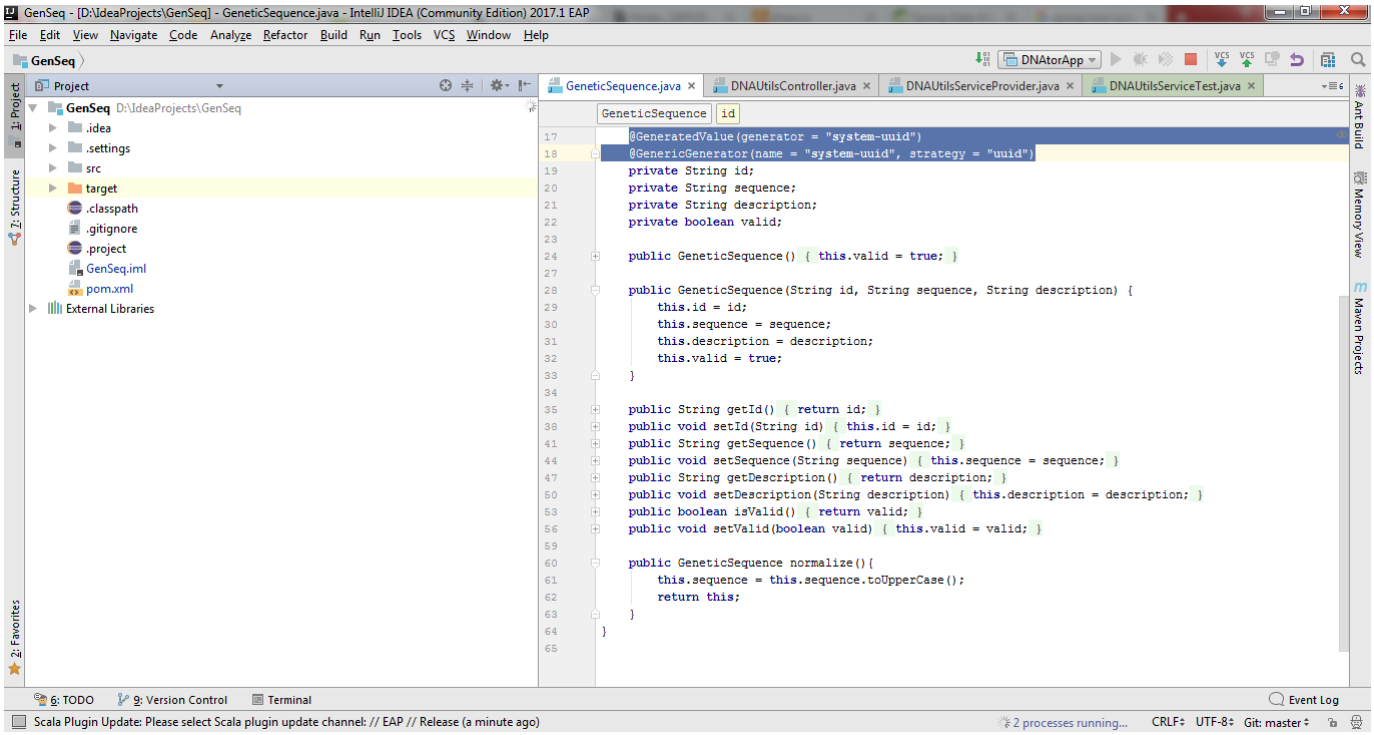

Рисунок 3.5 — Інтерфейс середовища IntelliJ IDEA

3.4.3. Visual Studio Code

Середовище Visual Studio Code [13] — текстовий редактор, розроблений у компанії Mirosoft, який досить часто використовується для написання додатків на мові JavaScript (рисунок 3.6).

Даний текстовий редактор залишається досить легким з точки зору навантаження на оперативну пам'ять та процесор, при цьому надаючи можливість встановлення різноманітних доповнень для ефективнішої розробки програмного продукту.

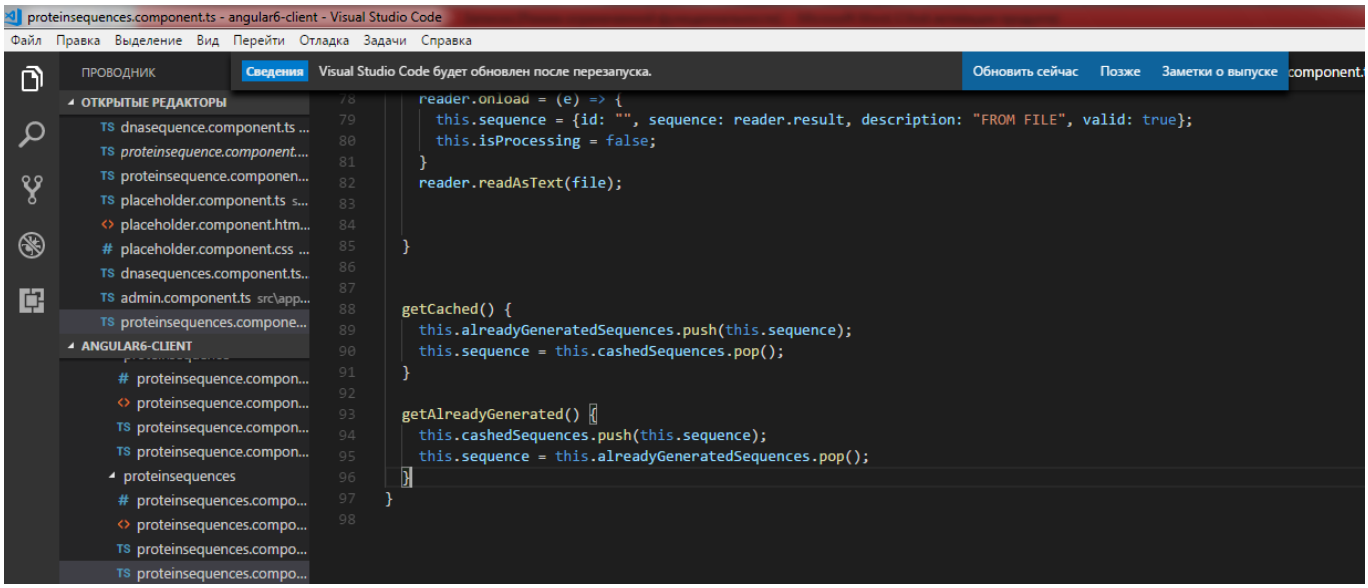

Рисунок 3.6 — Інтерфейс середовища Visual Studio Code

3.5. Висновки до розділу

В розділі було описано основні програмні засоби розробки, що були використані для побудови системи, зазначено їх переваги над аналогами.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Розроблена система складається з 3 відносно незалежних складових, структура яких подібна між собою, але має відмінності:

— бази даних;

— серверної частини з реалізацією алгоритмів обробки та аналізу;

— клієнтської частини, що відповідає за взаємодію з кінцевими користувачами продукту та зручне подання результатів роботи алгоритмів.

4.1. Структура бази даних

Екземпляр MongoDB, який слугує сховищем даних для системи, розміщений на кластері хмарного сервісу Amazon WebServices.

База даних системи містить 3 колекції:

— dnasequences (послідовності ДНК);

— rnasequences (послідовності РНК);

— proteinsequences (протеїни).

Об’єкти кожної з колекцій містять 4 поля:

— id (унікальний ідентифікатор): текст;

— sequence (послідовність у вигляді рядку з латинських літер): текст;

— description (опис): текст;

— valid (показник, що послідовність не містить помилок): логічне значення.

Використання однакового формату даних для колекцій послідовностей кожного типу забезпечує наскрізну уніфікацію потоку даних у програмній системі, одночасно відокремлюючи їх відповідно до специфіки предметної області.

4.2. Серверна частина системи

Сервер системи побудований з використанням шаблону “Модель — Представлення — Контролер”.

Задача редагування бази даних та додаткові інструменти відмежовані у дві окремі підсистеми.

Параметри сховища даних знаходяться у файлах dbinfo.txt та application.properties.

Взаємодія серверу з базою даних відбувається за допомогою фреймворку Spring Data (рисунок 4.1).

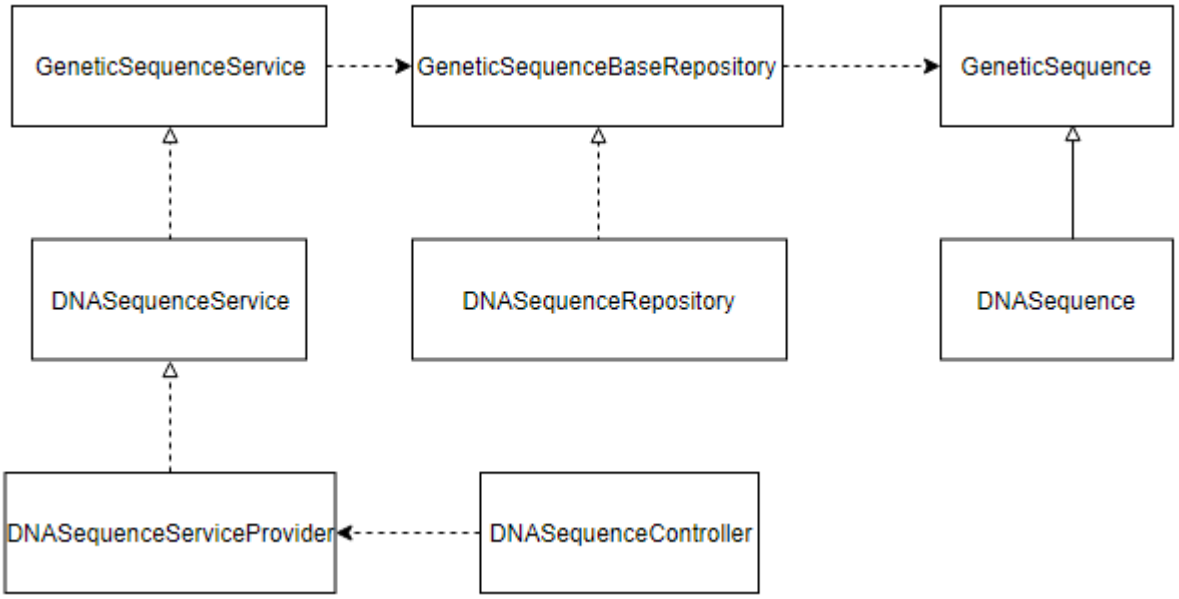

Рисунок 4.1 — Архітектура підсистеми взаємодії з базою даних для ДНК послідовностей

На вищому рівні абстракції існують клас GeneticSequence (абстракна сутність генетичної послідовності) та інтерфейси GeneticSequenceBaseRepository (методи взаємодії з базою даних для нащадків GeneticSequence, автоматично створені Spring Data) і GeneticSequenceService (адаптація результатів взаємодії з базою даних до подальших потреб системи).

Структура класу GeneticSequence містить наступні поля:

— private String id;

— private String sequence;

— private String description;

— private boolean valid.

Кожне поле є приватним, тобто не може бути змінене безпосередньо за межами класу, що їх містить. Для доступу до полів використовуються публічні методи доступу.

Абстрактний клас GeneticSequence не містить анотації сутності, оскільки неможливо створити його екземпляри для збереження у сховищі, однак поле ідентифікатора має анотацію @Id, яка наслідується конкретними нащадками класу.

Інтерфейс GeneticSequenceService містить лише три методи, які стосуються усіх конкретних імплементацій сервісів сховища даних послідовностей:

— void setGeneticSequence(GeneticSequence sequence);

— void setGeneticSequence(String id, GeneticSequence sequence);

— void removeGeneticSequence(String id).

Дані методи не повертають значення та приймають на вхід ідентифікатори або екземпляри генетичних послідовностей, здійснюючи операції їх додавання до бази, зміни або видалення.

Для кожного з типів послідовностей створюються нащадки вище зазначених абстрактних класів та інтерфейсів. Конкретні класи сервісів (як DNASequenceServiceProvider для ДНК послідовностей) динамічно зв’язуються з відповідними контролерами (як DNASequenceController). Контролери сприймають запити від клієнтської частини, ініціюють роботу сервісів та повертають результат у форматі JSON.

Наприклад, клас DNASequenceController має анотації @RestController та @RequestMapping(“/dnasequences”), тобто обробляє результати за адресами, що починаються на URL “/dnasequences”.

Сервіс DNASequenceService динамічно зв’язується з екземпляром контролеру за допомогою анотації @Autowired.

Контролер містить ряд методів для окремих адрес web-інтерфейсу:

— public List<DNASequence> getDNASequences();

— public List<DNASequence> getSimilarDNASequences (@RequestBody DNASequence dnaSequence);

— public DNASequence getDNASequence(@PathVariable String id);

— public void setDNASequence(@RequestBody DNASequence dnaSequence);

— public void updateDNASequence(@PathVariable String id, @RequestBody DNASequence dnaSequence);

— public void removeDNASequence(@PathVariable String id).

Анотація @PathVariable дозволяє оперувати параметрами GET-запитів, в той час як анотація @RequestBody відповідає за обробку вмісту тіла POST- або PUT-запиту.

Метод getSimilarDNASequences (@RequestBody DNASequence dnaSequence) принципово відрізняється від інших, оскільки не є простою операцією взаємодії з базою даних та є функціонально ближчим до методів, визначених у сервісі DNAUtilsService, однак повертає послідовності ДНК зі сховища і не може використовуватися самостійно, що сприяло його віднесенню до методів загального сервісу.

Необхідно зазначити складність алгоритму пошуку подібних послідовностей до заданої.

Спочатку задана послідовність підготовлюється до основного алгоритму (видаляються всі зайві символи). У ході основного алгоритму відбуваються ітеративні проходження списком ДНК-послідовностей бази та перевірки на входження для окремих фрагментів заданої послідовності. На кожній ітерації кількість таких приблизно рівних фрагментів на одиницю більша від логарифму з основою 4 від номеру ітерації за порядком.

Порівняння для кожного фрагменту виконуються паралельно. В разі успіху, послідовність з бази заноситься до множини. Використання структури даних множини дозволяє залишати лише унікальні збігання.

Даний метод використовує такі можливості мови програмування Java версії 8 як ітератори, Stream API та анонімні функції для застосування ефективного паралелізму.

Інша підсистема серверної частини займається реалізацією алгоритмів, специфічних для предметної області (рисунок 4.2).

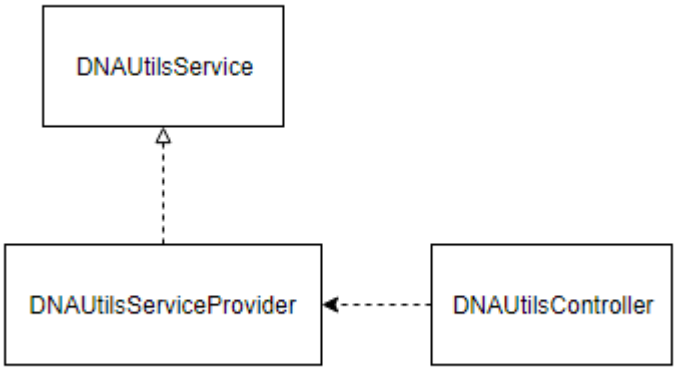

Рисунок 4.2 — Архітектура підсистеми додаткового функціоналу для ДНК послідовностей

Дана підсистема має простішу структуру, оскільки для неї відсутня пряма взаємодія з базою даних. Сервіс та контролер виконують ті ж задачі, що їх аналоги у першій підсистемі.

Зокрема, DNAUtilsService визначає інтерфейс методів редактору web-інтерфейсу.

Метод DNASequence complement(DNASequence dnaSequence) отримує на вхід послідовність ДНК та повертає її комплемент, тобто замінює усі входження елементів ADENINE на THYMINE, CYTOSINE на GUANINE і навпаки.

Метод Pair<Double, Double> measureBasePairPercentages(DNASequence dnaSequence) повертає пару раціональних чисел для послідовності ДНК, що містить відсоткове відношення пар елементів ADENINE / THYMINE до пар CYTOSINE / GUANINE.

Метод DNASequence mutate(DNASequence dnaSequence, int percentage) виконує найважчу з точки зору продуктивності обчислення операцію випадкової заміни певного відсотку нуклеотидних основ у послідовності та повертає відповідний мутований результат.

Метод DNASequence filter(DNASequence dnaSequence) видаляє з ДНК-послідовності усі символи, що не є коректними нуклеотидами і, у випадку наявності малих латинських літер для їх позначення, замінює їх відповідними великими.

Метод DNASequence random(int length) не приймає на вхід ДНК-послідовність, однак створює нову, випадкову, що має вказану довжину. Потрібно відмітити, що web-інтерфейс не дозволяє передавати серверу занадто важкі для обчислення задачі, тому генерація може виконуватися для довжин, що не перевищують число 300000.

Метод RNASequence transcript(DNASequence dnaSequence) проводить транскрипцію (специфічне перетворення) ДНК-послідовності у РНК-послідовність.

Останній метод, визначений у інтерфейсі сервісу, ProteinSequence translate(DNASequence dnaSequence) виконує специфічне перетворення трансляції ДНК у протеїнову послідовність. Трансляція містить у собі етапи фільтрації та транскрипції, отже одні методи сервісу викликають інші.

4.3. Клієнтська частина системи

Клієнтська частина система побудована з використанням фреймвору Angular 6 на мові програмування TypeScript з використанням мов розмітки HTML та CSS і системи керування модулями node.js (рисунок 4.3).

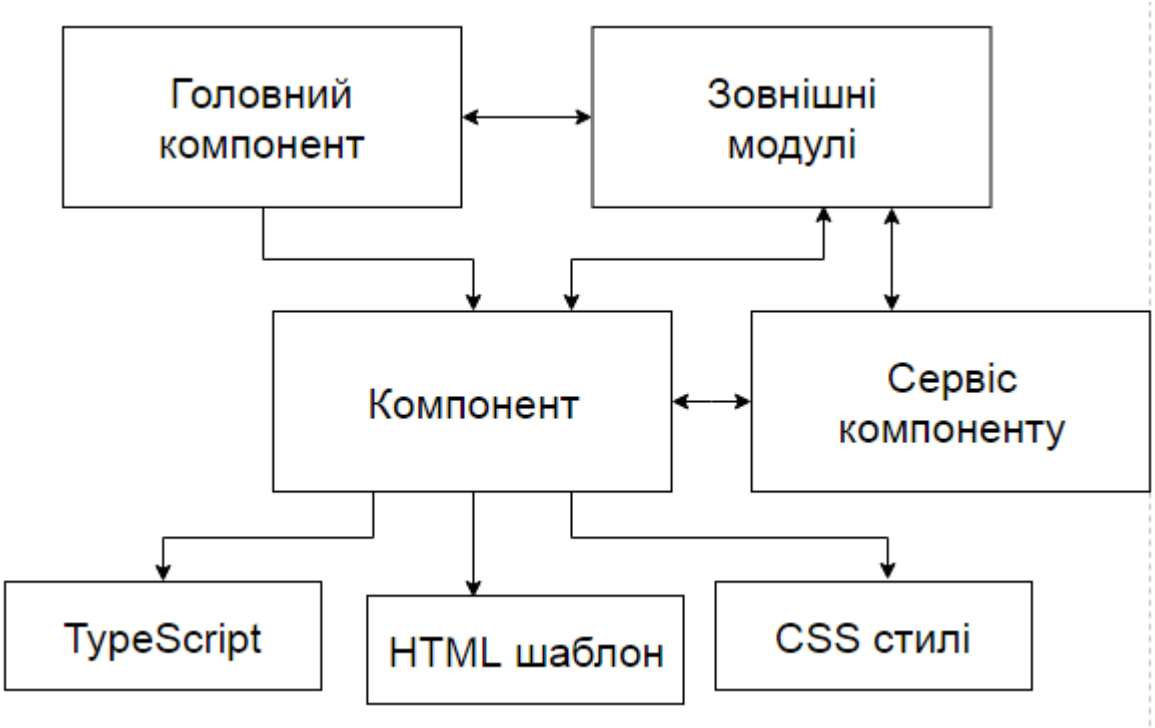

Рисунок 4.3 — Архітектура клієнтської частини системи

В основу архітектури клієнтської частини системи покладено принципи компонентного програмування.

Головний компонент сайту складається з інших компонентів, які теж, в свою чергу, можуть містити менші компоненти. Якщо компонент взаємодіє зі сервером, він має окремий сервіс, який виконує асинхронні запити до серверної частини та повертає результат, коли він стає доступним. Кожен компонент містить окремі файли для програмного коду, розмітки та стилів. Зовнішні модулі можуть включатися як до сервісів, так і до компонентів, щоб надати їм додатковий функціонал. Наприклад, було використано модуль ClipBoard, що реалізує операцію копіювання тексту послідовності при натисненні.

Модуль роутингу для застосунку співвідносить URL-адреси web-інтерфейсу із компонентами, що відображаються при переході на них кінцевого користувача.

У системі є наступні компоненти web-сторінок:

— PlaceholderComponent для відображення вмісту головної сторінки з посиланнями на зовнішні web-ресурси;

— DnasequencesComponent для відображення списку ДНК-послідовностей та редактора;

— DnasequenceComponent для відображення окремої ДНК-послідовності та редактора;

— RnasequencesComponent для відображення списку РНК-послідовностей та редактора;

— RnasequenceComponent для відображення окремої РНК-послідовності та редактора;

— ProteinsequencesComponent для відображення списку протеїнових послідовностей та редактора;

— ProteinsequenceComponent для відображення окремої протеїнової послідовності та редактора;

— AdminComponent для відображення сторінки адміністратора системи, який має можливості редагування вмісту сховища даних.

Компоненти виконують асинхронні запити до сервісів.

У складі системи є наступні сервіси:

— general;

— dnasequences;

— rnasequences;

— proteinsequences.

Запити до серверу виконуються через проксі-конфігурацію, визначену у файлі proxy.conf.json (рисунок 4.4).

```
{
  "/api/*": {
    "target": "http://localhost:8080",
    "secure": false,
    "pathRewrite": {"^/api" : ""}
  }
}
```

Рисунок 4.4 — Конфігурація взаємодії між клієнтською та серверною частинами системи у файлі proxy.conf.json

Конфігурація задає також формат URL-адреси, яку використовують сервіси для виконання HTTP-запитів до серверу.

4.4. Висновки до розділу

Отже, було описано основні принципи побудови програмної системи та конкретні приклади створення її елементів, акцентуючи увагу на застосованих шаблонах проектування.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

5.1. Системні вимоги та інсталяція

Для запуску системи на сервері спочатку необхідно переконатися у виконанні наступних вимог:

— мінімум 2 гігабайти оперативної пам’яті;

— наявність встановленого Java SDK 8 або його новішої версії.

Розробка та тестування програмного забезпечення відбувалися у середовищах різних операційних систем, що дозволяє гарантувати високу степінь універсальності створеного програмного рішення та можливість його запуску та коректного функціонування як для системи Windows, так і для Unix-орієнтованих операційних систем.

Щоб відбулося підключення до екземпляру MongoDB, необхідно ввести IP-адресу серверу до списку допустимих підключень кластеру.

Запуск серверної частини можна виконати командою “java -jar GenSeq.jar” у командному рядку операційної системи.

За умови успішного запуску серверної частини, порт 8081 надає точку доступу для засобу Spring Actuator, виконання запитів до якого дозволяє отримати актуальні дани про стан системи.

Запуск клієнтської частини виконується командою “npm run” у директорії проекту.

5.2. Сценарій роботи користувача з системою

Головна сторінка web-системи (рисунок 5.1) містить перелік посилань на інформаційні сайти біоінформатичної тематики, посилання на тематичні сторінки сайту (для кожного типу послідовностей) та форму для входу адміністратора.

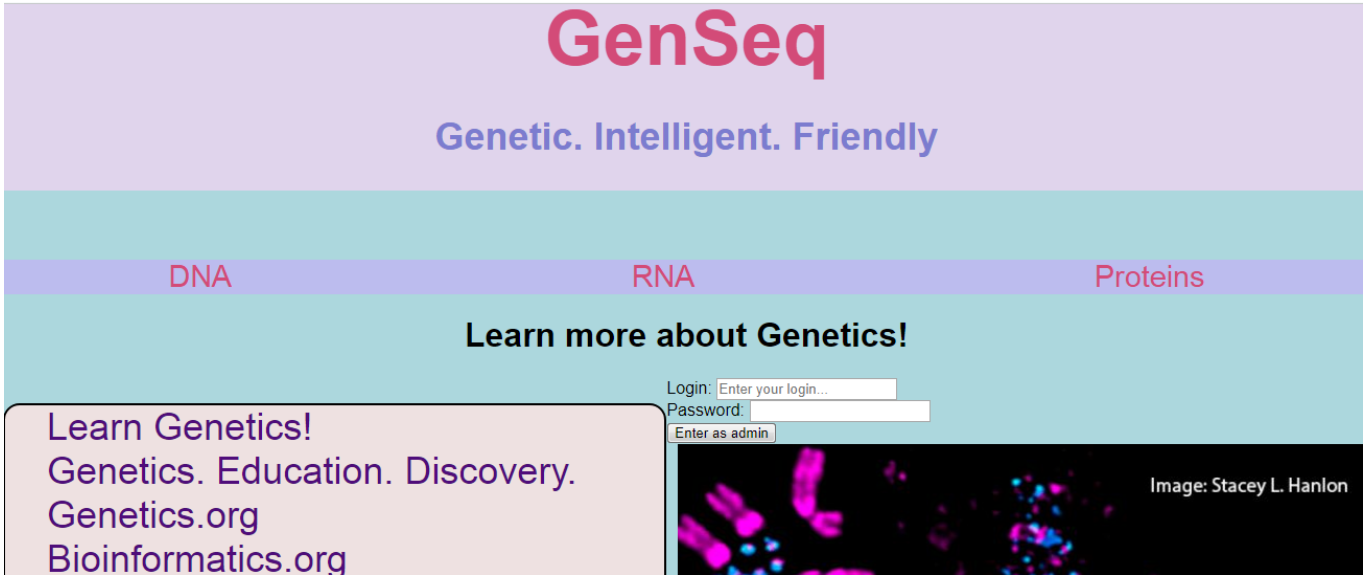

Рисунок 5.1 — Головна сторінка

Оскільки web-інтерфейс повинен зацікавити кінцевого користувача, головна сторінка насамперед пропонує ознайомлення з біоінформатикою як наукою, посилаючись на відомі Інтернет-ресурси.

При переході за посиланням на сторінку конкретного типу послідовностей користувач має можливість переглянути послідовності цього типу, які є у базі даних системи, та скористатися зручним редактором (рисунок 5.2).

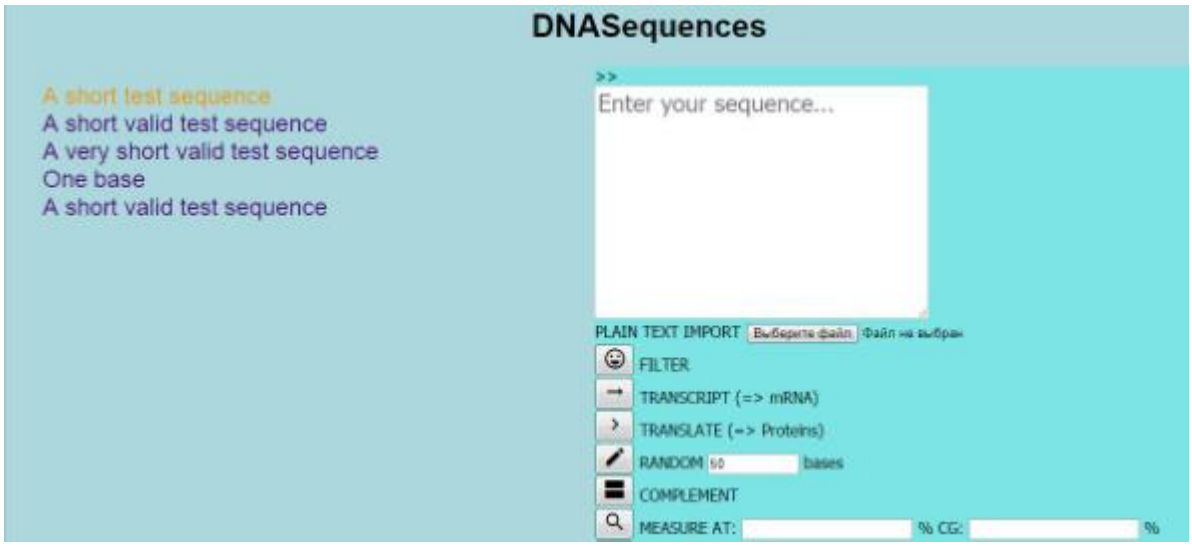

Рисунок 5.2 — Сторінка ДНК-послідовностей

Написи, що супроводжують окремі кнопки редактора, чітко та виразно передають зміст операцій. Активно використовується підсвічення у разі вибору елементів зі списку. В інтерфейсі домінують кольори подібних фізичних характеристик, щоб покращити досвід роботи користувача із системою.

При виборі конкретної послідовності користувач може переглянути її вміст та, натиснувши на неї, скопіювати його до буферу обміну (рисунок 5.3).

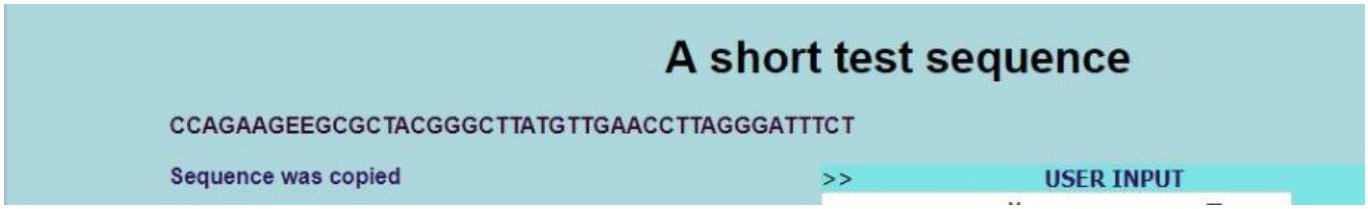

Рисунок 5.3 — Копіювання вмісту ДНК-послідовності

Редактор списку послідовностей містить на один інструмент більше, а саме засіб знаходження подібних послідовностей до заданої у базі даних системи.

Спільні для сторінок списку та окремої послідовності функції редактора включають

— завантаження послідовності з файлу, що має спеціально визначений формат, тобто дані складаються з рядку опису та безпосередньо послідовності;

— фільтрації послідовності;

— транскрипції ДНК до РНК;

— трансляції ДНК у протеїни (підтримується трансляція однієї послідовності ДНК у декілька протеїнів, якщо це можливо);

— генерації випадкової послідовності заданої довжини (не більше 300000);

— знаходження комплементу ДНК;

— вимірювання відношення пар нуклеотидів у ДНК-послідовності;

— мутації послідовності для заданого значення відсотку (не більше 2% для достатньо великих послідовностей, щоправда більші мутації практично не зустрічаються у природі).

З головної сторінки можна увійти до системи у якості адміністратора (рисунок 5.4). Для цього необхідно ввести правильні логін та пароль.

Система не передбачає можливості створення персональних кабінетів для користувачів, оскільки має іншу специфіку. Однак може бути багато адміністраторів, кожен з яких матиме свої персональні авторизаційні дані.

В будь-якому випадку, особливості авторизації регулюватимуться особливостями політики використання програмної системи.

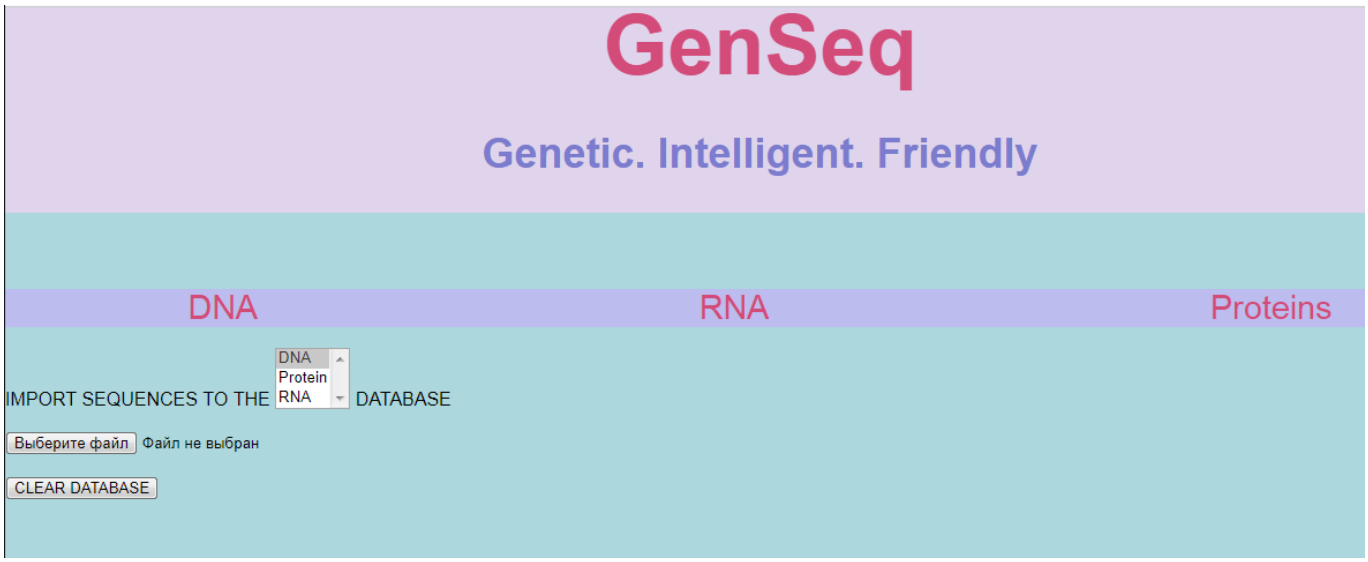

Рисунок 5.4 — Сторінка адміністратора

Адміністратор може завантажувати послідовності до бази даних системи та видаляти їх звідти. Для цього потрібно обрати тип послідовностей та завантажити зовнішній файл, що повинен містити послідовності одного типу у форматі, де рядок опису конкретного екземпляру чергується з його текстовим значенням.

5.3. Висновки до розділу

Було описано системні вимоги для запуску програмної системи, загальні інструкції по роботі з нею для кінцевого користувача та використані рішення зі сфери web-дизайну.

ВИСНОВКИ

У ході виконання дипломної роботи було створено мережевий додаток, що містить базовий функціонал для редагування та первинного аналізу генетичних послідовностей ДНК, РНК та протеїнів.

Система містить тимчасове сховище даних для збереження послідовностей кожного з даних типів.

Створений функціонал дозволяє виконувати задачі

— видалення помилкових символів з послідовності;

— транскрипції і трансляції ДНК;

— генерації випадкової послідовності для вказаної кількості символів;

— вимірювання відсоткового співвідношення пар нуклеотидів у ДНК;

— мутації послідовності;

— пошуку подібних послідовностей у базі тощо.

Планується розширення базового функціоналу, зокрема, із залученням технологій графічної візуалізації.

Система може бути інтегрована до більш складних системних комплексів або використовуватися самостійно, в тому числі для популяризації біоінформатичного підходу генетичної інженерії та вивчення молекулярних процесів у живих клітинах в цілому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rui Jiang, Xuegong Zhang, Michael Q. Zhang. Basics of Bioinformatics — Published by Tsinghua University Press, 2013 — P. 1-25

2. Будова клітин прокаріот і еукаріот — Острів знань [Електронний ресурс] — Режим доступу: http://shkola.ostriv.in.ua/publication/code-C0EA0CF8C63C/list-B65BB05F26.

3. The Sequence Manipulation Suite [Електронний ресурс] — Режим доступу: https://www.bioinformatics.org/sms2/.

4. Cybertory Home Page [Електронний ресурс] — Режим доступу: http://attotron.com/cybertory/.

5. BLAST: Basic Local Alignment Search Tool [Електронний ресурс] — Режим доступу: https://blast.ncbi.nlm.nih.gov/Blast.cgi.

6. The most popular database for modern apps | MongoDB [Електронний ресурс] — Режим доступу: https://www.mongodb.com.

7. Maven – Welcome to Apache Maven [Електронний ресурс] — Режим доступу: https://maven.apache.org.

8. Spring Boot [Електронний ресурс] — Режим доступу: https://spring.io/projects/spring-boot.

9. Spring Data [Електронний ресурс] — Режим доступу: https://spring.io/projects/spring-data.

10. Angular [Електронний ресурс] — Режим доступу: https://angular.io.

11. Spring Tools 4 [Електронний ресурс] — Режим доступу: https://spring.io/tools.

12. IntelliJ IDEA [Електронний ресурс] — Режим доступу: https://www.jetbrains.com/idea/.

13. Visual Studio Code [Електронний ресурс] — Режим доступу: https://code.visualstudio.com.

ДОДАТОК 1

Web-система для редагування та аналізу генетичних послідовностей

Специфікація

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ51197_19Б

Аркушів 2

Київ – 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ТІ 51197_19Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ТІ 51197_19Б 12-1	/dnaprocessing/*.java	Модулі серверної частини розробленої системи
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ТІ 51197_19Б 12-2	app/admin app/dnasequence app/dnasequences app/proteinsequence app/proteinsequences app/rnasequence app/rnasequences app/component.ts app/services	Модулі клієнської частини розробленої системи

ДОДАТОК 2

Сервіс утиліт для редагування та аналізу ДНК-послідовностей

Лістинг програмного модулю

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ51197_19Б 12-1

Аркушів 5

Київ – 2019

```
package edu.dnaprocessing.utils.dna;
//Підключення сторонніх бібліотек для генерації випадкових чисел
import java.util.Random;
//Підключення інших складових систем
import edu.dnaprocessing.sequence.protein.ProteinSequence;
import edu.dnaprocessing.sequence.rna.RNASequence;
import edu.dnaprocessing.utils.protein.ProteinUtilsService;
import org.springframework.data.util.Pair;
import org.springframework.stereotype.Service;

import edu.dnaprocessing.sequence.dna.DNASequence;
import edu.dnaprocessing.sequence.dna.DNAUnknownBaseException;
//Оголошення сервісу
@Service
public class DNAUtilsServiceProvider implements DNAUtilsService {
//Конкретна реалізація інтерфейсу
//Метод для фільтрації ДНК
	public DNASequence filter(DNASequence dnaSequence) {
		String sequence =
				prepareSequence(dnaSequence.getSequence());
		StringBuilder filteredSequence = new StringBuilder();
		String filteringPattern = new String(BASES);
		for(char possibleBase : sequence.toCharArray()){
			if(filteringPattern.contains(""+possibleBase))
				filteredSequence.append(possibleBase);
		}
		dnaSequence.setSequence(filteredSequence.toString());
		return dnaSequence;
	}
	//Метод генерації випадкової ДНК-послідовності
	public DNASequence random(int length) {
		Random random = new Random();
		DNASequence newSequence = new DNASequence();
		newSequence.setDescription(RANDOM_GENERATION_MARKER);
		StringBuilder sequenceBuilder = new StringBuilder();
		for(int i = 0; i < length; i++){
			sequenceBuilder.append(randomBase(random.nextInt(BASES.length +
i)));
		}
		newSequence.setSequence(sequenceBuilder.toString());
		return newSequence;
	}
//Метод знаходження комплементної ДНК
	public DNASequence complement(DNASequence dnaSequence) {
```

```
            String originalSequence = prepareSequence(dnaSequence.getSequence());
            StringBuilder complementSequenceBuilder = new StringBuilder();
            try{
                for(int i = 0; i < originalSequence.length(); i++){

        complementSequenceBuilder.append(complementBase(originalSequence.charAt(i))
);
                }
                dnaSequence.setSequence(complementSequenceBuilder.toString());
            }
            catch(DNAUnknownBaseException e){
                dnaSequence.setSequence(e.getMessage() + originalSequence);
                dnaSequence.setValid(false);
            }
                return dnaSequence;
        }
        //Метод вимірювання AT / CG співвідношення для послідовності
        public Pair<Double, Double> measureBasePairPercentages(DNASequence
dnaSequence) {
            int atCount = 0, cgCount = 0;
            char [] baseArray =
prepareSequence(dnaSequence.getSequence()).toCharArray();
            for(char base : baseArray){
                if(base == ADENINE || base == THYMINE)
                    atCount++;
                else
                    if(base == CYTOSINE || base == GUANINE)
                        cgCount++;
                    else
                        return Pair.of(0.0, 0.0);
            }
            double atPercentage = atCount * 100.0 / baseArray.length;
            double cgPercentage = cgCount * 100.0 / baseArray.length;
            return Pair.of(atPercentage, cgPercentage);
        }
        //Метод мутації ДНК заміною бази для вказаного відсотку нуклеотидів
        public DNASequence mutate(DNASequence sequence, int percentage) {
            StringBuilder sequenceBuilder = new
StringBuilder(prepareSequence(sequence.getSequence()));
            Random random = new Random();
            double countOfMutations = sequenceBuilder.length()*percentage*1.0/100;
            for(int i = 0; i < countOfMutations; i++){
                int randomIndex = CODON_LENGTH +
random.nextInt(sequenceBuilder.length() - 2*CODON_LENGTH);
```

```
                    char originalBase = sequenceBuilder.charAt(randomIndex);
                    boolean mutationDone = false;
                    do{
                    char mutatedBase = randomBase(randomIndex - CODON_LENGTH);
                    if(originalBase != mutatedBase){
                        sequenceBuilder.setCharAt(randomIndex, mutatedBase);
                        mutationDone = true;
                    }
                    } while(!mutationDone);
                }
                sequence.setSequence(sequenceBuilder.toString());
                return sequence;
        }
//Метод транскрипції ДНК
        public RNASequence transcript(DNASequence dnaSequence){
           RNASequence rnaSequence = new RNASequence();
           rnaSequence.setSequence(dnaSequence.getSequence().replace(THYMINE,
URACIL));
           rnaSequence.setDescription("mRNA: " + dnaSequence.getDescription());
           return rnaSequence;
    }
//Метод трансляції ДНК
    public ProteinSequence translate(DNASequence dnaSequence){
                boolean inFrame = false;
                int openReadingFrameCount = 0;
                ProteinSequence proteinSequence = new ProteinSequence();
                RNASequence rnaSequence = transcript(dnaSequence);
                StringBuilder proteinSequenceBuilder = new StringBuilder();
                try{
                    for(int i = 0; i < rnaSequence.getSequence().length(); i++){
                        if(!inFrame &&
                                isStartCodon("" +
rnaSequence.getSequence().charAt(i)
                                  + rnaSequence.getSequence().charAt(i+1)
                                  + rnaSequence.getSequence().charAt(i+2))){
                                inFrame = true;
                                openReadingFrameCount++;

        proteinSequenceBuilder.append(ProteinUtilsService.OPEN_READING_FRAME_
ABBREVIATION);

        proteinSequenceBuilder.append(openReadingFrameCount);
```

```
        proteinSequenceBuilder.append(ProteinUtilsService.OPEN_READING_FRAME_S
TART);

        proteinSequenceBuilder.append(ProteinUtilsService.METHIONINE);
                                        i += 3;
                                }
                                if(inFrame){
                                        String codon = "" + rnaSequence.getSequence().charAt(i)
                                                        + rnaSequence.getSequence().charAt(i+1)
                                                        + rnaSequence.getSequence().charAt(i+2);
                                        if(isStopCodon(codon)){
                                                inFrame = false;

        proteinSequenceBuilder.append(ProteinUtilsService.OPEN_READING_FRAME_
END);
                                        }
                                        else{

        proteinSequenceBuilder.append(translateCodon(codon));
                                        }
                                        i += 2;
                                }
                        }
                }
                catch(IndexOutOfBoundsException ex){
                        System.out.println("Translation: " + ex.getMessage());
                }
                proteinSequence.setSequence(""+proteinSequenceBuilder);
                return proteinSequence;
        }


        private char randomBase(int randomNumber){
                return BASES[randomNumber % BASES.length];
        }

}
```

ДОДАТОК 3

Сервіс утиліт для редагування та аналізу ДНК-послідовностей

Опис програмного модулю

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТІ51197_19Б 12-2

Аркушів 8

Київ – 2019

АНОТАЦІЯ

Додаток надає короткий функціональний та структурний опис реалізації сервісу утиліт для ДНК-послідовностей, що знаходиться у серверній частині системи у якості одного з найважливіших модулів цілого програмного комплексу.

Методи сервісу отримують на вхід послідовності ДНК та, за необхідності, деяку числову інформацію, реалізуючи стандартні алгоритми редагування та первинного аналізу.

Модуль було створено у середовищі Spring Tool Suite на мові програмування Java.

ЗМІСТ

-4-

ЗАГАЛЬНІ ВІДОМОСТІ

Програмний код модулю міститься в межах класу DNAUtilsServiceProvider, який реалізує методи, оголошені у інтерфейсі DNAUtilsService.

Модуль відіграє важливу роль у функціонуванні серверної частини системи, задовольняючи запити контролера DNAUtilsController.

Програма була написана мовою Java з використанням потужного фреймворку Spring.

-5-

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Сервіс реалізує наступний функціонал:

— фільтрації послідовності ДНК;

— генерації випадкової послідовності;

— знаходження комплементу;

— виконання випадкової мутації зразка;

— транскрипції до РНК;

— трансляції до протеїну;

— вимірювання відношення базового співвідношення між основами.

-6-

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Модуль реалізовано у формі класу, який анотовано як сервіс для підключення до контролеру. Даний клас реалізує відповідний інтерфейс, що містить оголошення та короткий опис наступних методів:

1) DNASequence complement(DNASequence dnaSequence);
2) Pair<Double,Double>measureBasePairPercentages(DNASequence dnaSequence);
3) DNASequence mutate(DNASequence dnaSequence, int percentage);
4) DNASequence filter(DNASequence dnaSequence);
5) DNASequence random(int length);
6) RNASequence transcript(DNASequence dnaSequence);
7) ProteinSequence translate(DNASequence dnaSequence).

-7-

ТЕХНІЧНІ ЗАСОБИ, ЩО ВИКОРИСТОВУВАЛИСЯ

Програмний код модулю було написано у середовищі розробки Spring Tool Suite та пізніше відредаговано у середовищі IntelliJ IDEA.

До версіонування модулю було залучено систему контроля версій Git.

Робоча версія класу сервісу компілюється у байт-код віртуальною машиною мови програмування Java за умови запуску серверної частини на виконання.

-8-

ВХІДНІ ТА ВИХІДНІ ДАНІ

Вхідними даними є:

— ДНК-послідовність;

— максимальний допустимий відсоток мутацій;

— довжина випадково обчисленої послідовності.

Вихідними даними є:

— перетворена ДНК;

— РНК;

— протеїн;

— відомості про співвідношення основ у зразку.